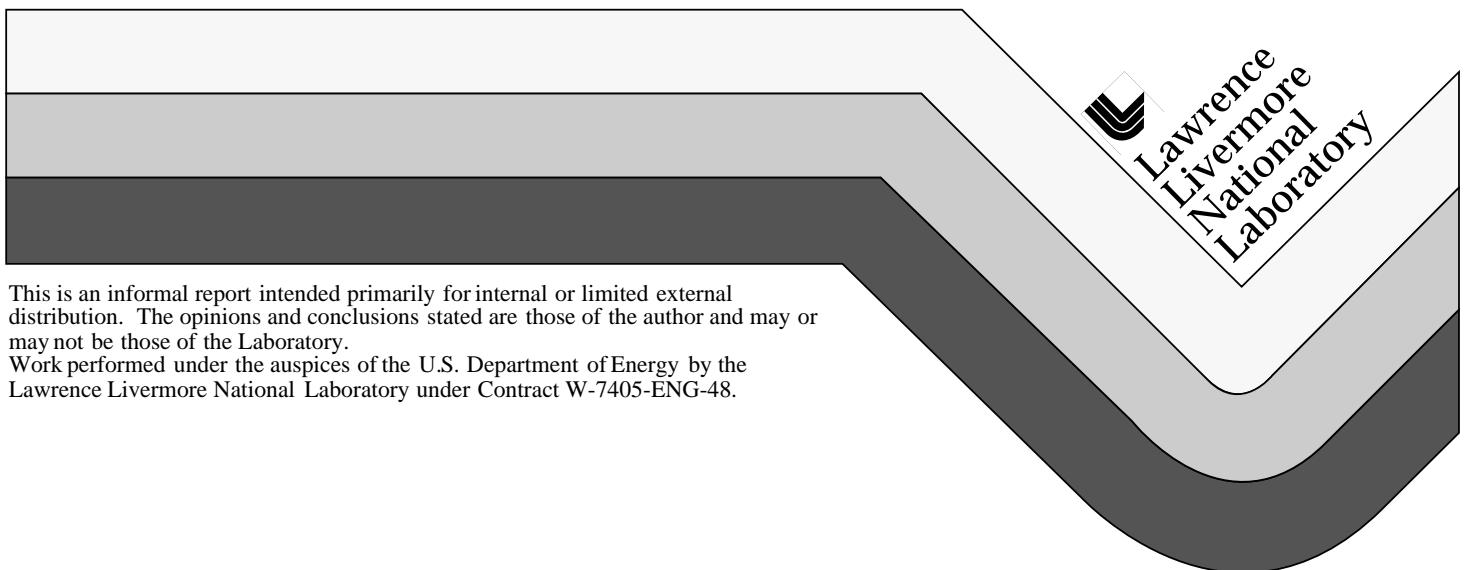


Explanation of How to Run the Global Local Optimization Code (GLO) to Find Surface Heat Flux

Vivek Sahai
Salvador Aceves
Werner Stein

March 1999



This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the Laboratory.

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This report has been reproduced
directly from the best available copy.

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information
P.O. Box 62, Oak Ridge, TN 37831
Prices available from (423) 576-8401

Available to the public from the
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd.,
Springfield, VA 22161

**Explanation of How to Run the
Global Local Optimization Code (GLO)
to Find Surface Heat Flux**

by

Vivek Sahai
Salvador Aceves
Werner Stein

University of California
Lawrence Livermore National Laboratory
P.O. Box 808, L-140
Livermore, CA 94551

Introduction

From the evaluation[1] of the inverse techniques available, it was determined that the Global Local Optimization Code[2] can determine the surface heat flux using known experimental data at various points in the geometry. This code uses a whole domain approach in which an analysis code (such as TOPAZ2D or ABAQUS) can be run to get the appropriate data needed to minimize the heat flux function. This document is a compilation of our notes on how to run this code to find the surface heat flux. First, the code is described and the overall set-up procedure is reviewed. Then, creation of the configuration file is described. A specific configuration file is given with appropriate explanation. Using this information, the reader should be able to run GLO to find the surface heat flux.

Description of GLO and Overall Setup Approach

The GLO code [2] uses whole domain specification to find the surface heat fluxes Mathematically, whole domain specification[3-4]can be described as follows:

Find the Heat Flux Function $E(q)$

where $q = \{ q_1, q_2, q_i, \dots, q_{nvariables} \}$

such that $q_{\text{minimum}} < q_i < q_{\text{maximum}}$

where the values of q_i minimize the figure of merit which is $\sum(T_{\text{experiment}} - T_{\text{calc}})^2$
(the summation occurs over all time steps)

The optimization strategy that works best in GLO is the quasi-newton method with a finite difference gradient search technique. The iterative equations that describe the Newton's method are given below(j is the iteration number, $E(q)$ is the heat flux function, and q represents the array of heat flux variables, and nvariables is the total number of variables):

$$\nabla^2 E(q_j) \cdot \Delta q = -\nabla E(q_j)$$

$$q_{j+1} = q_j + \Delta q$$

To set-up GLO to do the calculations, a configuration file is created. This file contains the commands to run the numerical analysis code(either TOPAZ2D or ABAQUS) to calculate the temperatures based on the current iteration values of the heat flux. Then ORION is used as a post-processing code that processes the t2plot file created by TOPAZ2D. ORION creates a file that contains the cooling curve data for the nodes that correspond to the actual experimental thermocouple positions. This configuration file also contains the details of the parameters that are to be optimized (heat flux variables). It sets the details of the iteration loop and how to calculate the figure of merit that the code uses to do the optimization.

Finally, the overall procedure used to get the heat flux results is described below :

- (a) create sub-files (which are placed into the configuration file) containing the experimental thermocouple data,
- (b) create the sub-file which runs the numerical analysis code,
- (c) make the glo-configuration file,
- (d) run the GLO analysis (for two iterations),
- (e) check the *.log file to see if everything is working,
- (f) run the GLO analysis, and finally,
- (g) analyze the results by comparing with experimental data

The key step in the procedure in running GLO is the creation of the configuration file. This file is described in the next section.

GLO Configuration File (xxx.GCF)

The configuration file is very important for running the GLO code. This file contains the following elements:

- (a) GLO commands (such as glo-put, glo-get, loop, glo-file, and end glo-file)
- (b) Analysis Code Run Commands (such as the TOPAZ2D/ABAQUS execution command)
- (c) Commands Used to Extract the Numerical Data (ORION execution command)
- (d) Optimization Strategy (Quasi-Newton technique)
- (e) Optimization Parameters (The names of the heat flux variables, minimums, maximums)
- (f) Analysis Code Input File (TOPAZ2D input file/ ABAQUS input file)
- (g) Figure of Merit Calculations
- (h) Command File listing ORION commands to print out time vs. temperature data from the t2plot file created by the TOPAZ2D code (ORION Command File)
- (i) Experimental Data used in the Figure of Merit Calculation.

The configuration file consists of several sections(sub-files) which contain the above elements. Each section begins with the command “glo-file = name of the file” and ends with the command “end-glo file”.

The first section is called the “iloc” sub-file. This sub-file contains the information of where the GLO executable file resides, the iteration analysis loop, optimization strategy, and the heat flux curve parameter names with their minimum, maximum, and initial values (elements (a) through (e) are placed here). The iteration loop lets the GLO code place the values of the heat flux in the numerical analysis code input file, run the post-processor, and then call the file to do the figure of merit calculation (done with the glo-get command).

The second section is a slightly modified form of the numerical analysis code input file. This sub-file contains the details of the numerical analysis options, material properties, mesh, node numbers and locations, element definitions, heat flux boundary condition location, and heat flux curve details. In this sub-file, a glo-format command is used in the details of the heat flux curve which tells GLO where to put the values of the heat flux needed as a boundary condition to run the analysis code(element (f)).

The third section is the actual figure of merit calculation. This sub-file is given the name, calc. In this sub-file, the variables are defined and the names of the files are given from which to read both the experimental and numerical values. Then a do loop calculates the figure of merit and finally the value is printed out (element (g)).

The fourth section is the actual command sequence to print out the numerical values from the analysis code (element (h)). Finally, the last set of sub-files contain the experimental thermocouple data (one set of time vs. temperature data in each file)(element i). The specific glo-commands used in these files are further described in the GLO manual[2]. The best way to understand the configuration file is through the example given in the last section of this document.

When creating the GLO configuration file, the following points must be kept in consideration:

I. Make sure the number of significant figures is enough for the finite difference calculations; if not, change the value of the dx_grad parameter. In our work, to determine the surface heat fluxes of the Saturn Gear Blank, we recommend dx_grad be set to 0.001.

II. In the figure of merit calculations (glo-file named calc), make sure the number of points in the experimental data equals the number of points generated by the post-processing code, ORION. To avoid the need for interpolation, make sure the comparison between the numerical and experimental data occur at the same time step.

In the example section, a specific GLO gcf file is given which will emphasize the information given above. To understand more detail of what each command in GLO does, please refer to the GLO users manual[2].

Running the GLO code and the Files Created During the Run

Once the gcf file is created, the command line to run GLO on a SGI IRIX type system is simply the following:

```
glo xxx(gcf > out &
```

where xxx(gcf is the name of the gcf file, and out is the file where the screen output is directed to. After hitting the return key to start the run, a number(nnnn) is given which is the process id of the GLO calculations, If one needs to kill the calculations, use the following command:

```
kill -9 nnnn
```

During the run, the following files are created which give the status of the run:

- (a) **out** which contains the screen output (from TOPAZ2D and ORION for example and information from the GLO controller with the figure of merit)
- (b) **xxx.log** which contains the progress of GLO along with any error messages or other troubles during the run along with the commands performed during the run for each iteration
- (c) **local.l1** which documents the values of the variables of each run and the commands from the loop executed during each run, the type of run (i.e. gradient evaluation or newton step) and list the figure of merit during the run.
- (d) **local.u1** which lists the iteration number and value of the figure of merit
- (e) **varvals** which contains the current values of the parameters to be optimized (ie. heat flux values), and finally,
- (f) **glo-nnnn.recover** which is used to restart GLO if the system crashes during a run.

To use the recover file, be sure not to make any changes to the gcf file and use the command:

```
glo -recover glo-nnnn.recover xxx(gcf > out &
```

During the GLO run, each iteration's work is stored in a separate directory, i.e. ./pyyyy/

where yyyy is the iteration number. This directory contains the analysis code input file with the current values of heat flux for that iteration, and all the command files used to extract the numerical output, as well as the numerical output, etc. To save disk space, one can use “rm” commands in the gcf to delete unnecessary files. It is strongly recommended that after the run, all these directories be removed except for the last iteration number directory!

Files to Examine After the GLO Run

The values of all the heat flux parameters are contained in the **solution.0000** and **varvals** files. In the **solution.0000** file, the final value of the figure of merit and the last iteration number is also given. The **local.u2** file contains the values of the figure of merit for each iteration. To understand the solution, one should examine the last iteration number directory. The final version of the analysis code input file is contained in the **./pxxx/** directory where xxxx is the number of the last iteration. One should run this file to compare the results of the predicted temperatures with the experimental data. Examples of how these files look like are given in the example problem in the next section. Again for space management, one can delete all the files in the **./pyyy/** directories **before** the last iteration xxxx.

Example Problem

A two-dimensional square cavity 10 units long by 10 units high has a uniform constant heat flux applied to the back face which causes a rise in temperature from a 0 degrees uniform initial temperature. Known temperature data exists for the mid-point position on the back face. The other three faces are assumed to be insulated. The objective is to use GLO to find the value of the heat flux.

The diagram of the situation is shown below:

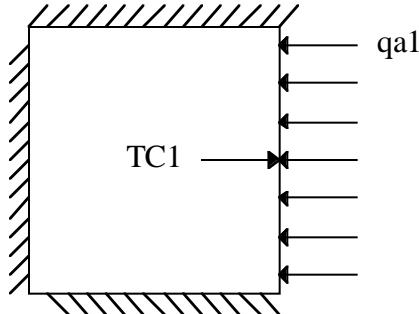


Figure 1. Sketch of Square Cavity with a Uniform Heat Flux applied on the back surface and known temperature data located at the mid-point on the back surface.

An outline of the calculational procedure that GLO uses is given in Figure 2. The file names on the left are the ones that are created by GLO from the information in the gcf file. The file names below each box are the ones created by the code. The “fom” stands for the figure of merit variable calculated by the **calc** file. This value is used by the optimizer to determine whether to continue iterating or print out the results. The variable name “qa1” is the value of the heat flux parameter which is specified in the analysis code input file (**ttest1.ref**) that is being optimized by GLO. The value of the heat flux to be found is given the variable name qa1. As seen from the outline, first GLO reads the entire gcf file, then

puts the initial value of “qa1” in the file ttest1.ref at the position indicated by the glo-format command, and creates the file ttest1. Then ttest1 is used in TOPAZ to do the transient

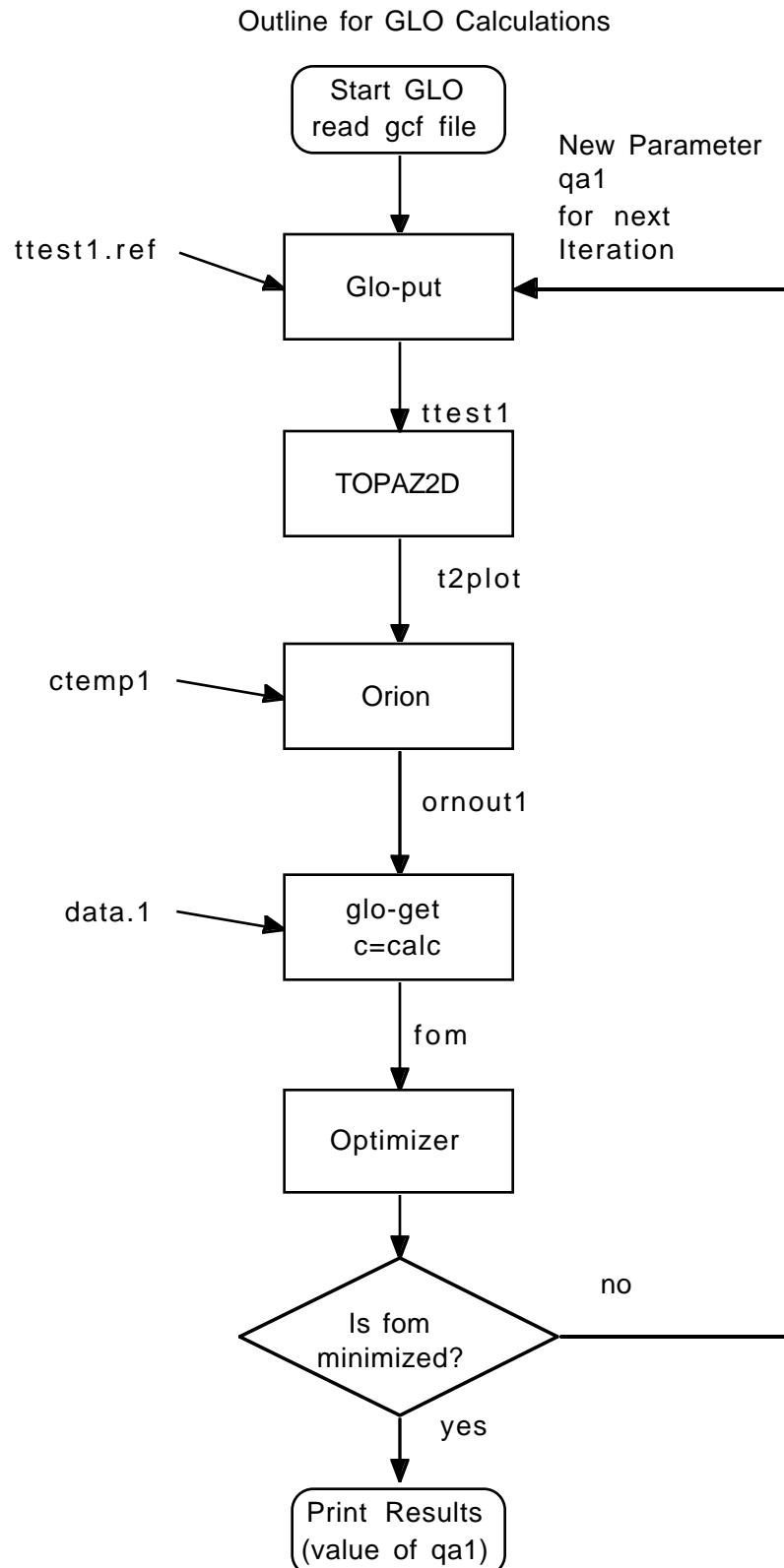


Figure 2. Flow Chart Showing Iteration Loop of GLO Calculations

heat transfer analysis and the results from the TOPAZ analysis are placed in the output file t2plot. Then ORION is used to get the cooling curve data at the node corresponding to the location of TC1. The commands to get this information are contained in the sub-file in the gcf named ctemp1. ORION puts this information in the file ornout which is moved to ornout1. Then this information from ornout1 and the experimental data data.1 is read into the sub-file named calc which determines the figure of merit whose value is stored in the variable named “fom”. This value is used in the optimizer which then determines the next value of qa1 for the next iteration. If the figure of merit is optimized, then the results are printed in the names of the files listed in the previous section.

The GLO configuration file used to solve this problem using the heat transfer analysis code TOPAZ2D and the post-processing code ORION is given in Table 1. There are total of five sub-files or sections that make up this gcf. Please look at the # or \$ comment cards which explain what each line does. The words in bold face give specific details of each section or line of the configuration file.

The resulting files created by GLO (i.e. local.l1, local.l2, local.u2, and solution.0000) are given in the Appendix. The results show that a total of five iterations were performed. The first iteration was an initial evaluation followed by 2 Newton steps and 2 gradient evaluations. The values of the figure of merit was quickly minimized to value less than 1. The final value of the optimized heat flux was 0.09975.

To analyze the results, the GLO calculated heat flux value can be checked by doing a direct simulation(i.e. a TOPAZ2D transient heat conduction simulation with the GLO predicted value of heat flux used as a boundary condition). The comparison between the calculated temperatures(using the optimized value of qa1) and the experimental data is shown in Figure 3. As one can see, the agreement between the GLO results and the known data is very good.

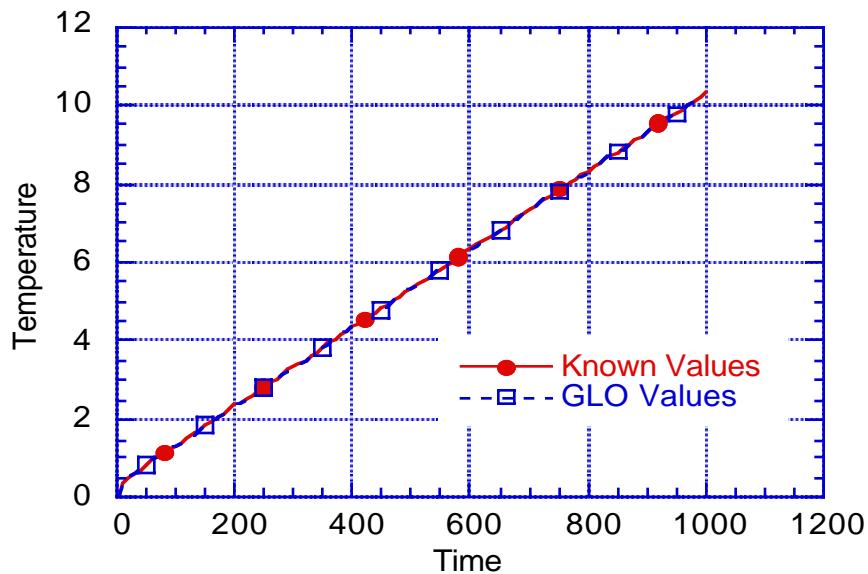


Figure 3. GLO calculated Temperatures vs. Known Data

Summary

This document has given the reader the necessary information to set-up the GLO code to perform whole domain optimization to find the surface heat flux. Our work with GLO has shown the usefulness of this approach. For example, previous work at LLNL has shown the ability of the code not only predict the surface heat flux but the variation of the heat flux along the surface[1]. The ability of GLO to calculate surface heat fluxes for a horizontally quenched Saturn Gear Blank in non-agitated oil[5] has been demonstrated.

In conclusion, a summary of the procedure for setting up and using the GLO code is as follows:

- (a) set up the experimental data files
- (b) set up the numerical analysis input file
- (c) use this information in the creation of a GLO configuration file
- (d) run GLO
- (e) analyze the results.

References

- [1] M.Y. Lin, M. Murphy, A. Shapiro, W. Stein, "Numerical Methods for Inverse Heat Transfer Analysis", Lawrence Livermore National Laboratory Report, September 1997.
- [2] R. Matzke, "GLO User's Manual", Lawrence Livermore National Laboratory, 1995.
- [3] G.S. Dulikravich, T.J. Martin, "Inverse Shape and Boundary Condition Problems and Optimization in Heat Conduction", in Advances in Numerical Heat Transfer Volume 1 editors W. J. Minkowycz and E.M. Sparrow, Talyor and Francis, Washington DC 1997 pg. 409.
- [4] J.E. Dennis Jr., Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice Hall, Inc. NJ, 1983.
- [5] V. Sahai, Analysis of an Oil Quenched Gear Blank, Presentation at First Components Demonstration Meeting, Kissimmee, Florida February 23-25, 1998.

Table 1. GLO Configuration File(square.gcf) used to Solve Square Cavity Problem

```

# A # comments out a whole line
# A $ comments till the end of the line
#####
# square.gcf $ name of this gcf file
#####
optimizer = /usr/people/sahai/GLO/bin/local i=iloc $location of glo-executable and
# name of first sub-file which contains main loop
#####
ncpu = 1      $ Max number of cpus available
#####
# Section Number 1 (sub-file named iloc)
#####
glo-file = iloc
$ initialize the non-default optimization code settings
settings
max_pfn 0102 $ run loop this many times
parallel 1     $ run one problem at a time in parallel
dx_grad 0.001   $ relative finite difference gradient step
end settings

# define the local optimization problem analysis loop
loop $ this starts the loop (outline given in Figure 2)
glo-put i=ttest1.ref o=ttest1
# the above command takes the sub-file ttest1.ref(the input file that is specified given
# in the sub-file named ttest1.ref within this gcf file) places the value of the heat flux
# variable qa1 into ttest1 and produces the input file, ttest1, for the TOPAZ2D
# analysis. Each iteration produces a ttest1 file that always contains the latest value
# of qa1
topaz2d.sgi i=ttest1      $ execution line for TOPAZ2D
# TOPAZ2D produces plot files that contain the calculational output
# The name of this file is t2plot
orion.sgi g=t2plot c=ctemp1 $ execution line for orion to post-process results
# ctemp1 is a sub-file given in this gcf. ctemp1 contains commands that create the
# time temperature data for the node at the location of TC1 (see Figure 1) This data is
# stored in the file name ornout
mv ornout ornout1      $ moves analysis results to a file named ornout1 to be read by calc
rm dig.ps*              $ unix commands to remove unnecessary files
rm t2plot
rm tprint
rm t2dump*
rm newfile
glo-get c=calc
# name of glo sub-file (given in this gcf) which does objective function analysis
end loop $ used to end the optimization analysis loop

$ define the optimization strategy
strategy
local nlqpeb $ for Quasi-Newton with finite difference gradient
#
# Heat Flux Parameter to be optimized is defined here.
# parameter name, initial value, lowerbound, upperbound, 0.0
#

```

```

parameter qa1 0.5e0 0.0001 1.0e3 0.0
# qa1 represents the name of the value of the unknown heat flux that has to be
# optimized. This parameter is placed in sub-file named ttest1.ref.
# This line specifies, for qa1, the initial value 0.5, minimum value 0.0001 and
# maximum value 1.0e3. For this type of optimization procedure each of these
# parameter lines must end in 0.0
end strategy $after all parameters are named use this command to end the
# optimization strategy description
run
# always use the run command to end the iloc file so that glo will start running!
end glo-file
# always use glo-file = name to begin a file and end glo-file to end a file
#####
# Section Number 2 - TOPAZ2D input file (sub-file named ttest1.ref)
#####
glo-file = ttest1.ref
* This is a TOPAZ2D input file and therefore in this file every line
* beginning with a * represents a comment line
* TOPAZ2D Problem Definition: Card 1
*
* Field
* [1] Title
*
topaz input file: square problem for glo run
*
* TOPAZ2D Problem Definition: Card 2
*
* Field
* [1] Number of Materials      [ 7] Number of data pairs
* [2] Number of nodes          [ 8] Number of slidelines
* [3] Number of elements       [ 9] Total number of slave nodes
* [4] Type of geometry        [10] Total number of master nodes
* [5] Bandwidth minimization   [11] Solution Method
* [6] Number of function curves[12] Conjugate gradient convergence tolerance
*
      1    121   100     2     1     1     2     0     0     0     0     3 1e-4
*
* TOPAZ2D Problem Definition: Card 3
*
* Field
* [1] Number of elements with [ 6] Number of radiation
*      internal heat generation boundary conditions
* [2] Number of nodes with    [ 7] Number of enclosure
*      non-zero initial temp.  radiation segments
* [3] Number of nodes with    [ 8] Number of radiation bands
*      temp. boundary conditions[ 9] Number of emissivity vs. wavelength curves
* [4] Number of flux boundary [10] Radiation calculation type
*      condition segments      [11] Number of special internal boundary elements
* [5] Number of convection    [12] Number of bulk nodes
*      boundary condition      [13] Number of bulk node segments
*      segments                 [14] Number of chemical reaction elements
*                                [15] Number of fluid flow elements
*
      0     0     0    10     0     0     0     0     0     0     0     0     0
*
* TOPAZ2D Problem Definition: Card 4
* Field
* [1] Analysis type           [5] Node and element dump time
* [2] Time step code           interval for printing
* [3] Node and element dump    [6] Node and element dump time
*      step interval for        for post-processing
*      printing                 [7] Number of steps between
* [4] Node and element dump    restart dumps

```

```

*      step interval for          [8] Newmark parameter
*      post-processing
*
1     0    200    10     0 5.000E-01 1.000E+00 5.000E+01
*
* TOPAZ2D Problem Definition: Card 5
*
* Field           Field
* [1] Initial problem time   [5] Maximum time step size
* [2] Final problem time    [6] Maximum temperature change in each time step
* [3] Initial time step size above which time step will be decreased
* [4] Minimum time step size [7] Time step control parameter
* NOTICE: we are using fixed time steps(times must match experimental
* data)! Time step is 1 sec,from card 4[4] we are saving every 10 steps!
* Therefore, the total number of datapoints will be 101.
0.0000E+00 1000.0     1.0     0.0     0.0     0.0     0.5
* TOPAZ2D Problem Definition: Card 6
*
* Field           Field
* [1] Problem type       [3] Maximum number of equilibrium
* [2] Maximum number of   iterations between stiffness
*      stiffness matrix    matrix reformations
*      reformations per time [4] Convergence tolerance
*      step                  [5] Divergence control parameter
*
0     1    100.0000E+001.0000E+00
*material name, number, and properties
cube      1     1     1.0     0.0     0.0     0     0.0
        1.0     1.0
* node number increment x and y location
1  0.0  0.000000  0.000000
2  0.0  1.000000  0.000000
3  0.0  2.000000  0.000000
4  0.0  3.000000  0.000000
5  0.0  4.000000  0.000000
6  0.0  5.000000  0.000000
7  0.0  6.000000  0.000000
8  0.0  7.000000  0.000000
9  0.0  8.000000  0.000000
10 0.0  9.000000  0.000000
11 0.0  10.000000 0.000000
12 0.0  0.000000  1.000000
13 0.0  1.000000  1.000000
14 0.0  2.000000  1.000000
15 0.0  3.000000  1.000000
16 0.0  4.000000  1.000000
17 0.0  5.000000  1.000000
18 0.0  6.000000  1.000000
19 0.0  6.999999  1.000000
20 0.0  8.000000  1.000000
21 0.0  8.999999  1.000000
22 0.0  10.000000 1.000000
23 0.0  0.000000  2.000000
24 0.0  1.000000  2.000000
25 0.0  2.000000  2.000000
26 0.0  3.000000  2.000000
27 0.0  4.000000  2.000000
28 0.0  5.000000  2.000000
29 0.0  6.000000  2.000000
30 0.0  7.000000  2.000000
31 0.0  7.999999  2.000000
32 0.0  9.000000  2.000000
33 0.0  10.000000 2.000000
34 0.0  0.000000  3.000000

```

35	0.0	1.000000	3.000000
36	0.0	2.000000	3.000000
37	0.0	3.000000	3.000000
38	0.0	4.000000	3.000000
39	0.0	5.000000	3.000000
40	0.0	6.000000	3.000000
41	0.0	7.000000	3.000000
42	0.0	8.000000	3.000000
43	0.0	9.000000	3.000000
44	0.0	10.000000	3.000000
45	0.0	0.000000	4.000000
46	0.0	1.000000	4.000000
47	0.0	2.000000	4.000000
48	0.0	3.000000	4.000000
49	0.0	4.000000	4.000000
50	0.0	5.000000	4.000000
51	0.0	6.000000	4.000000
52	0.0	7.000001	4.000000
53	0.0	8.000000	4.000000
54	0.0	9.000000	4.000000
55	0.0	10.000000	4.000000
56	0.0	0.000000	5.000000
57	0.0	1.000000	5.000000
58	0.0	2.000000	5.000000
59	0.0	3.000000	5.000000
60	0.0	4.000000	5.000000
61	0.0	5.000000	5.000000
62	0.0	6.000000	5.000000
63	0.0	7.000000	5.000000
64	0.0	8.000000	5.000000
65	0.0	9.000000	5.000000
66	0.0	10.000000	5.000000
67	0.0	0.000000	6.000000
68	0.0	1.000000	6.000000
69	0.0	2.000000	6.000000
70	0.0	3.000000	6.000000
71	0.0	4.000000	6.000000
72	0.0	5.000000	6.000000
73	0.0	6.000000	6.000000
74	0.0	7.000000	6.000000
75	0.0	8.000000	6.000000
76	0.0	9.000000	6.000000
77	0.0	10.000000	6.000000
78	0.0	0.000000	7.000000
79	0.0	1.000000	7.000000
80	0.0	2.000000	7.000000
81	0.0	3.000000	7.000000
82	0.0	4.000000	6.999999
83	0.0	5.000000	7.000000
84	0.0	6.000000	7.000000
85	0.0	7.000000	7.000000
86	0.0	8.000000	7.000000
87	0.0	9.000000	7.000000
88	0.0	10.000000	7.000000
89	0.0	0.000000	8.000000
90	0.0	1.000000	8.000000
91	0.0	2.000000	7.999999
92	0.0	3.000000	8.000000
93	0.0	4.000000	8.000000
94	0.0	5.000000	8.000000
95	0.0	6.000000	8.000000
96	0.0	7.000000	8.000000
97	0.0	7.999999	7.999999
98	0.0	9.000001	8.000000

99	0.0	10.000000	8.000000		
100	0.0	0.000000	9.000000		
101	0.0	1.000000	9.000001		
102	0.0	2.000000	9.000000		
103	0.0	3.000000	9.000000		
104	0.0	4.000000	9.000000		
105	0.0	5.000000	9.000000		
106	0.0	6.000000	9.000000		
107	0.0	7.000000	9.000000		
108	0.0	8.000000	9.000001		
109	0.0	9.000001	9.000001		
110	0.0	10.000000	9.000000		
111	0.0	0.000000	10.000000		
112	0.0	1.000000	10.000000		
113	0.0	2.000000	10.000000		
114	0.0	3.000000	10.000000		
115	0.0	4.000000	10.000000		
116	0.0	5.000000	10.000000		
117	0.0	6.000000	10.000000		
118	0.0	7.000000	10.000000		
119	0.0	8.000000	10.000000		
120	0.0	9.000000	10.000000		
121	0.0	10.000000	10.000000		
* element definitions, element number,					
* four nodes that make up the element and material number					
1	1	2	13	12	1
2	2	3	14	13	1
3	3	4	15	14	1
4	4	5	16	15	1
5	5	6	17	16	1
6	6	7	18	17	1
7	7	8	19	18	1
8	8	9	20	19	1
9	9	10	21	20	1
10	10	11	22	21	1
11	12	13	24	23	1
12	13	14	25	24	1
13	14	15	26	25	1
14	15	16	27	26	1
15	16	17	28	27	1
16	17	18	29	28	1
17	18	19	30	29	1
18	19	20	31	30	1
19	20	21	32	31	1
20	21	22	33	32	1
21	23	24	35	34	1
22	24	25	36	35	1
23	25	26	37	36	1
24	26	27	38	37	1
25	27	28	39	38	1
26	28	29	40	39	1
27	29	30	41	40	1
28	30	31	42	41	1
29	31	32	43	42	1
30	32	33	44	43	1
31	34	35	46	45	1
32	35	36	47	46	1
33	36	37	48	47	1
34	37	38	49	48	1
35	38	39	50	49	1
36	39	40	51	50	1
37	40	41	52	51	1
38	41	42	53	52	1
39	42	43	54	53	1

40	43	44	55	54	1
41	45	46	57	56	1
42	46	47	58	57	1
43	47	48	59	58	1
44	48	49	60	59	1
45	49	50	61	60	1
46	50	51	62	61	1
47	51	52	63	62	1
48	52	53	64	63	1
49	53	54	65	64	1
50	54	55	66	65	1
51	56	57	68	67	1
52	57	58	69	68	1
53	58	59	70	69	1
54	59	60	71	70	1
55	60	61	72	71	1
56	61	62	73	72	1
57	62	63	74	73	1
58	63	64	75	74	1
59	64	65	76	75	1
60	65	66	77	76	1
61	67	68	79	78	1
62	68	69	80	79	1
63	69	70	81	80	1
64	70	71	82	81	1
65	71	72	83	82	1
66	72	73	84	83	1
67	73	74	85	84	1
68	74	75	86	85	1
69	75	76	87	86	1
70	76	77	88	87	1
71	78	79	90	89	1
72	79	80	91	90	1
73	80	81	92	91	1
74	81	82	93	92	1
75	82	83	94	93	1
76	83	84	95	94	1
77	84	85	96	95	1
78	85	86	97	96	1
79	86	87	98	97	1
80	87	88	99	98	1
81	89	90	101	100	1
82	90	91	102	101	1
83	91	92	103	102	1
84	92	93	104	103	1
85	93	94	105	104	1
86	94	95	106	105	1
87	95	96	107	106	1
88	96	97	108	107	1
89	97	98	109	108	1
90	98	99	110	109	1
91	100	101	112	111	1
92	101	102	113	112	1
93	102	103	114	113	1
94	103	104	115	114	1
95	104	105	116	115	1
96	105	106	117	116	1
97	106	107	118	117	1
98	107	108	119	118	1
99	108	109	120	119	1
100	109	110	121	120	1

* heat flux boundary conditions

* node nos. of face,heat flux curve number, multiplication factors at each node

* the node numbers make up the face for which heat flux curve number 1 is applied

```

11 22          1 -1.0 -1.0
22 33          1 -1.0 -1.0
33 44          1 -1.0 -1.0
44 55          1 -1.0 -1.0
55 66          1 -1.0 -1.0
66 77          1 -1.0 -1.0
77 88          1 -1.0 -1.0
88 99          1 -1.0 -1.0
99 110         1 -1.0 -1.0
110 121        1 -1.0 -1.0

* Heat Flux Curve Definition for Curve Number 1 with curve title of q1.
* This is the location where glo inputs the latest value of the heat
* flux variable qal !! The glo-format command is the format
* specification to tell GLO how to input the value of qal which is being
* optimized
*curve identification name(q1), curve number, number of pairs of points
    q1 1 2
* time, value of qal
    0.000[glo-format "%10.4e"][%qal]
    1000.0[glo-format "%10.4e"][%qal]
* To specify flux for TOPAZ2D input, a time vs. heat flux function is
* used. The 10.4e represents the FORTRAN format specification for the
* heat flux value named qal. The %qal tells GLO to enter this value with
* the format specification of 10.4e. The 10.4e is the format requirement
* of TOPAZ2D.
end glo-file
#####
#Section Number 3 subfile named calc
# Calculation of Objective Function (sub-file named calc)
#####
glo-file = calc
# define cost files
file t_temp1 "ornout1", fortran ;
file e_temp1 "data.1", fortran ;
# t_temp1 is the name given to the ORION numerical output file ornout1 and e_temp1
# is the name of sub-file data.1 containing experimental data (data.1 is in this gcf)
# read fom's from each cost file
# define a 2xn matrix
var t1(:,2) = t_temp1/101 ;
var e1(:,2) = e_temp1/101 ;
# t1 and e1 are arrays that should contain 101 data sets of time and temperature
# The var command tells GLO: that t_temp1 and e_temp1 are files containing data
# to start reading the data into the second column of each array after the line with
# characters 101. We know that the ORION output starts after the number 101 since
# set up the TOPAZ2D analysis to print out 101 sets of time and temperature.
# Array t1 contains the numerical data (t_temp1) & e1 has experimental data
# (e_temp1)
# define initial fom value of 0.0
var fom = 0.0 ;
# fom is the variable that contains the figure of merit
# calculate figure of merit
# always make sure that all your output has 101 entries and coincide at the
# same time step
# do loop to calculate the fom
for i (1:101)
{
    fom=fom+(t1(i,2)-e1(i,2))*(t1(i,2)-e1(i,2)) ;
}
# print the total fom
print_fom fom ;
end glo-file

```

```

#####
# Section Number 4 sub-file named ctemp1
# This is the ORION batch commands file that creates the ornout sub-file that contains
# print out of the numerical output from TOPAZ2D (i.e. extract from t2plot the 101
# pairs of time and temperature data for node number 66 which corresponds to the
# location of the experimental data location TC1) (see Figure 1 for node location)
glo-file = ctemp1
2
phs2 nodes 1 66
gather
column
print
ntim 7 1 66
end
end glo-file
#####
# Section Number 5 sub-file named data.1
# Experimental Data TC1(GLO reads the values after the number 101 in line 1)
# This is where one inputs experimental data into GCF file. (101 pairs of data points)
glo-file = data.1
101
0.00000000000E+00 0.00000000000E+00
1.00000000000E+01 3.55887889862E-01
2.00000000000E+01 5.04599153996E-01
3.00000000000E+01 6.22270822525E-01
4.00000000000E+01 7.28785693645E-01
5.00000000000E+01 8.31125855446E-01
6.00000000000E+01 9.31982398033E-01
7.00000000000E+01 1.03227508068E+00
8.00000000000E+01 1.13243913651E+00
9.00000000000E+01 1.23248779774E+00
1.00000000000E+02 1.33248746395E+00
1.10000000000E+02 1.43243193626E+00
1.20000000000E+02 1.53236198425E+00
1.30000000000E+02 1.63235318661E+00
1.40000000000E+02 1.73228228092E+00
1.50000000000E+02 1.83218479156E+00
1.60000000000E+02 1.93223464489E+00
1.70000000000E+02 2.03213477135E+00
1.80000000000E+02 2.13219428062E+00
1.90000000000E+02 2.23208665848E+00
2.00000000000E+02 2.33215188980E+00
2.10000000000E+02 2.43204236031E+00
2.20000000000E+02 2.53210949898E+00
2.30000000000E+02 2.63199758530E+00
2.40000000000E+02 2.73183917999E+00
2.50000000000E+02 2.83193373680E+00
2.60000000000E+02 2.93177175522E+00
2.70000000000E+02 3.03186464310E+00
2.80000000000E+02 3.13169908524E+00
2.90000000000E+02 3.23179006577E+00
3.00000000000E+02 3.33161973953E+00
3.10000000000E+02 3.43170905113E+00
3.20000000000E+02 3.53153395653E+00
3.30000000000E+02 3.63161516190E+00
3.40000000000E+02 3.73141527176E+00
3.50000000000E+02 3.83148813248E+00
3.60000000000E+02 3.93128252029E+00

```

3.700000000000E+02	4.03135347366E+00
3.800000000000E+02	4.13114166260E+00
3.900000000000E+02	4.23121070862E+00
4.000000000000E+02	4.33099317551E+00
4.100000000000E+02	4.43106031418E+00
4.200000000000E+02	4.53083658218E+00
4.300000000000E+02	4.63090276718E+00
4.400000000000E+02	4.73067235947E+00
4.500000000000E+02	4.83073759079E+00
4.600000000000E+02	4.93047666550E+00
4.700000000000E+02	5.03014802933E+00
4.800000000000E+02	5.13023948669E+00
4.900000000000E+02	5.22990655899E+00
5.000000000000E+02	5.32999753952E+00
5.100000000000E+02	5.42966079712E+00
5.200000000000E+02	5.52975082397E+00
5.300000000000E+02	5.62941026688E+00
5.400000000000E+02	5.72949981689E+00
5.500000000000E+02	5.82915592194E+00
5.600000000000E+02	5.92924499512E+00
5.700000000000E+02	6.02884578705E+00
5.800000000000E+02	6.12889862061E+00
5.900000000000E+02	6.22850465775E+00
6.000000000000E+02	6.32855701447E+00
6.100000000000E+02	6.42816686630E+00
6.200000000000E+02	6.52821969986E+00
6.300000000000E+02	6.62783336639E+00
6.400000000000E+02	6.72735404968E+00
6.500000000000E+02	6.82743597031E+00
6.600000000000E+02	6.92692661285E+00
6.700000000000E+02	7.02700901031E+00
6.800000000000E+02	7.12651538849E+00
6.900000000000E+02	7.22659826279E+00
7.000000000000E+02	7.32611751556E+00
7.100000000000E+02	7.42614650726E+00
7.200000000000E+02	7.52560758591E+00
7.300000000000E+02	7.62564086914E+00
7.400000000000E+02	7.72512865067E+00
7.500000000000E+02	7.82516574860E+00
7.600000000000E+02	7.92464399338E+00
7.700000000000E+02	8.02400398254E+00
7.800000000000E+02	8.12407588959E+00
7.900000000000E+02	8.22347640991E+00
8.000000000000E+02	8.32355117798E+00
8.100000000000E+02	8.42288780212E+00
8.200000000000E+02	8.52290248871E+00
8.300000000000E+02	8.62229347229E+00
8.400000000000E+02	8.72158527374E+00
8.500000000000E+02	8.82164192200E+00
8.600000000000E+02	8.92093086243E+00
8.700000000000E+02	9.02099514008E+00
8.800000000000E+02	9.12030315399E+00
8.900000000000E+02	9.22029685974E+00
9.000000000000E+02	9.31960296631E+00
9.100000000000E+02	9.41881561279E+00
9.200000000000E+02	9.51886081696E+00
9.300000000000E+02	9.61808586121E+00
9.400000000000E+02	9.71814346313E+00

```

9.50000000000E+02 9.81732273102E+00
9.60000000000E+02 9.91730976105E+00
9.70000000000E+02 1.00165424347E+01
9.80000000000E+02 1.01156644821E+01
9.90000000000E+02 1.02157106400E+01
1.00000000000E+03 1.03148479462E+01
end glo-file
#####

```

Appendix. Results Files

A. solution.0000 file (gives number of cycles, final value of qa1, and fom)

```

#####
# local solution cycle 5      #
# parameter qa1      9.9750080093E-02 # p 1 5 g 0
# solution glo-fom    2.2600883066E-02 # f 1 5 g 0
#####

```

B. local.u2 file (gives the value of the fom for each iteration)

```

# fom
1 5.7858497519E+04
2 5.8003234021E+04
3 3.6089273876E+03
4 3.5728923923E+03
5 2.2600883066E-02

```

C. local.l1 file (for each iteration, it reproduces the commands done in each iteration, lists the value of the figure of merit, the value of qa1 used in each iteration and the type of iteration [initial run, newton step, gradient evaluation, etc.])

```

local      local-version:97.ad date:97/04/28 release:glo-970429a
#####
nlqp analysis - local # 1
nlqp state - initial run

parameter qa1      5.0000000000E-01
parameter lcycle          1
glo-put i=ttest1.ref o=ttest1
topaz2d.sgi i=ttest1
orion.sgi g=t2plot c=ctemp1
mv ornout ornout1
rm dig.ps*
rm t2plot
rm tprint
rm t2dump*
rm newfile
glo-get c=calc
#####
fom = 5.7858497519E+04 local 1

```

```

#####
##### nlqp analysis - local # 2
##### nlqp state - gradient evaluation

parameter qa1      5.0050000000E-01
parameter lcycle     2
glo-put i=ttest1.ref o=ttest1
topaz2d.sgi i=ttest1
orion.sgi g=t2plot c=ctemp1
mv ornout ornout1
rm dig.ps*
rm t2plot
rm tprint
rm t2dump*
rm newfile
glo-get c=calc
#####
fom =  5.8003234021E+04 local  2
#####
##### nlqp analysis - local # 3
##### nlqp state - newton step

parameter qa1      1.0000000000E-04
parameter lcycle     3
glo-put i=ttest1.ref o=ttest1
topaz2d.sgi i=ttest1
orion.sgi g=t2plot c=ctemp1
mv ornout ornout1
rm dig.ps*
rm t2plot
rm tprint
rm t2dump*
rm newfile
glo-get c=calc
#####
fom =  3.6089273876E+03 local  3
#####
**nlqpeb: jcycle,dp,dpmax:  3  0.00050  0.02000
#####
##### nlqp analysis - local # 4
##### nlqp state - gradient evaluation

parameter qa1      6.0000000000E-04
parameter lcycle     4
glo-put i=ttest1.ref o=ttest1
topaz2d.sgi i=ttest1
orion.sgi g=t2plot c=ctemp1
mv ornout ornout1
rm dig.ps*
rm t2plot
rm tprint
rm t2dump*
rm newfile

```

```

glo-get c=calc
#####
fom = 3.5728923923E+03 local 4
#####
nlqp analysis - local # 5
nlqp state - newton step

parameter qa1 9.9750080093E-02
parameter lcycle 5
glo-put i=ttest1.ref o=ttest1
topaz2d.sgi i=ttest1
orion.sgi g=t2plot c=ctemp1
mv ornout ornout1
rm dig.ps*
rm t2plot
rm tprint
rm t2dump*
rm newfile
glo-get c=calc
#####
fom = 2.2600883066E-02 local 5
#####
**nlqpeb: jcycle,dp,dpmx: 5 0.00010 0.02000

```

D. local.l2 file (gives in a summarized format some of the information given in local.l2 i.e. value of the figure of merit, the value of qa1 used in each iteration and the glo state)

```

local local-version:97.ad date:97/04/28 release:glo-970429a
#####
# solution glo-run local 1      #
# glo state initial run        #
# parameter qa1 5.0000000000E-01 # p 1 1
# solution glo-fom 5.7858497519E+04 # f 1 1
#####
# solution glo-run local 2      #
# glo state gradient evaluation #
# parameter qa1 5.0050000000E-01 # p 1 2
# solution glo-fom 5.8003234021E+04 # f 1 2
#####
# solution glo-run local 3      #
# glo state newton step         #
# parameter qa1 1.0000000000E-04 # p 1 3
# solution glo-fom 3.6089273876E+03 # f 1 3
#####
# solution glo-run local 4      #
# glo state gradient evaluation #
# parameter qa1 6.0000000000E-04 # p 1 4
# solution glo-fom 3.5728923923E+03 # f 1 4
#####
# solution glo-run local 5      #
# glo state newton step         #
# parameter qa1 9.9750080093E-02 # p 1 5
# solution glo-fom 2.2600883066E-02 # f 1 5

```

#####